WHAT IS CLAIMED IS:

1. In a computer system using an operating system to provide access to hardware resources, wherein said operating system provides access to said resources via a first source code component, a method of replacing said first source code component with a new source code component while said operating system remains active and while said operating system provides continual availability to applications of the hardware resources, the method comprising:

identifying references to said first source code component; and

replacing the identified references to said first source code with references to said new source code component.

2. A method according to Claim 1, wherein the method is implemented transparently to said applications.

3. A method according to Claim 1, wherein the method is scalable.

4. A method according to Claim 1, wherein said mechanism is divided across a multiprocessor system so that each processor can proceed independently.

5. A method according to Claim 1, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component;

transferring said quiescent state from the first code component to the new code component; and

after transferring said quiescent state, swapping the first code component with the new code component.

6. A method according to Claim 1, wherein the step of identifying references includes the steps of:

separating the first code component into objects; and

grouping said objects into a table, whereby references to said objects are entered in the table.

7. A method according to Claim 1, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component, without locking the first code component, by tracking active threads to the first code component;

transferring said quiescent state from the first code component to the new code component; and

after transferring said quiescent state, swapping the first code component with the new code component.

8. A method according to Claim 1, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component;

transferring the quiescent state from the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and

after transferring said quiescent state, swapping the first code component with the new code component.

9. A method according to Claim 1, wherein:

the step of identifying references includes the steps of

separating the first code component into objects, and

grouping said objects into a table, whereby references to said objects are entered in the table; and

the replacing step includes the steps of

establishing a quiescent state for the first code component;

transferring the quiescent state from the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and

after transferring said quiescent state, swapping the first code component with the new code component.

10. A system for swapping source code in a computer system including an operating system, said operating system including at least one source code component and providing continual availability to applications of hardware resources, the system comprising:

means for identifying, while said operating system is active and providing continual access to said resources, references to a first source code component of the operating system; and

means for replacing the identified references, while said operating system is active and providing continual access to said resources, to said first source code with references to a new source code component for the operating system.

11. A system according to Claim 10, wherein the system operates transparently to said applications and the system is scalable.

12. A system according to Claim 10, wherein said mechanism is divided across a multiprocessor system so that each processor can process independently.

13. A system according to Claim 10, wherein the means for replacing step includes:

means for establishing a quiescent state for the first code component;

means for transferring said quiescent state from the first code component to the new code component; and

means for swapping the first code component with the new code component after said quiescent state has been transferred.

14. A system according to Claim 10, wherein the means for identifying references includes:

means for separating the first code component into objects; and

means for grouping said objects into a table, whereby references to said objects are entered in the table.

15. A system according to Claim 10, wherein the means for replacing includes:

means for establishing a quiescent state for the first code component, without locking the first code component, by tracking active threads to the first code component;

means for transferring said quiescent state from the first code component to the new code component; and

means for swapping the first code component with the new code component after said quiescent state has been transferred.

16. A system according to Claim 10, wherein the means for replacing includes:

means for establishing a quiescent state for the first code component;

means for transferring the quiescent state from the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and

means for swapping the first code component with the new code component after said quiescent state has been transferred.

17. A method according to Claim 10, wherein:

the means for identifying references includes

i) means for separating the first code component into objects, and

ii) means for grouping said objects into a table, whereby references to said objects are entered in the table; and

the means for replacing includes

i) means for establishing a quiescent state for the first code component;

ii) means for transferring the quiescent state from the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and

iii) means for swapping the first code component with the new code component after said quiescent state has been transferred.

18. A program storage device, for use with a computer system including an operating system to provide access to hardware resources, wherein said operating system provides access to said resources via a first source code component, said program storage device being readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for replacing said first source code component with a new source code component while said operating system remains active and while said operating system provides continual availability to applications of said resources, the method steps comprising:

identifying references to said first source code component; and

replacing the identified references to said first source code with references to said new source code component.

19. A program storage device according to Claim 18, wherein the method is implemented transparently to said applications, the method is scalable, and said mechanism is divided across a multiprocessor system so that each processor can proceed independently.

20. A program storage device according to Claim 18, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component;

transferring said quiescent state from the first code component to the new code component; and

after transferring said quiescent state, swapping the first code component with the new code component.

21. A program storage device according to Claim 18, wherein the step of identifying references includes the steps of:

separating the first code component into objects; and

grouping said objects into a table, whereby references to said objects are entered in the table.

22. A program storage device according to Claim 18, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component, without locking the first code component, by tracking active threads to the first code component;

transferring said quiescent state from the first code component to the new code component; and

after transferring said quiescent state, swapping the first code component with the new code component.

23. A program storage device according to Claim 18, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component;

transferring the quiescent state from the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and

after transferring said quiescent state, swapping the first code component with the new code component.

24. A program storage device according to Claim 18, wherein:

the step of identifying references includes the steps of

i) separating the first code component into objects, and

ii) grouping said objects into a table, whereby references to said objects are entered in the table; and

the replacing step includes the steps of

i) establishing a quiescent state for the first code component;

ii) transferring the quiescent state from the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and

iii) after transferring said quiescent state, swapping the first code component with the new code component.